How to Customize a Multiplication Operation

There are at least two ways to consider a multiplication operation. The first way is to consider a multiplication to be a mathematical operation. Such an operation is a mental one and has a specific meaning and a pre-determined outcome. One cannot change such a mental operation without changing its fundamental meaning.

The second way to look at a multiplication operation is as a description of a machine or computer operation. A computer does not perform a multiplication, but merely performs a series of switching operations. The overall computer operation modeled as for instance a modulo-*n* multiplication has certain properties. The operation is closed (all inputs and outputs are from the same *n*-state group of elements); the operation is commutative ($a*b=b*a$); the operation is associative ($[a*b]*c=a*[b*c]=b*[a*c]$); there is a neutral element *e* so that $a*e=a$ for all states of *a*; each state *a* (except for one state *z*) has an inverse so that $a*a^{-1}=e$; there is a state *z* for which $a*z=z$ for all states of *a*. In general $e=1$ and $z=0$ in the mathematical interpretation of the operation '*'.

A Novel Number-theoretical Approach: the FLT

In order to customize '*' into '⊛' a transformation is performed. This transformation is illustrated as a transformation of an *n*-state switching table of an operation, which is called the Finite Lab-Transform (FLT). The FLT may also be modeled as a programmed rule. This transformation (FLT) is believed to be novel and not currently known in mathematical literature.

Inverters and Transformation: 5-state example

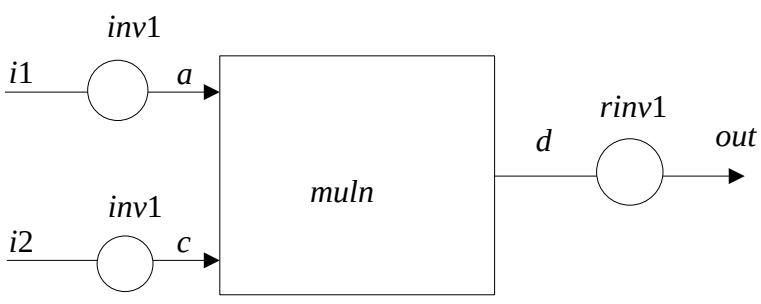
The FLT uses a device or computer operation that is called a reversible *n*-state inverter. An *n*-state element has one of *n* (different) states. For illustrative purpose, states will be assumed to be numbered from 0 to (n-1). A 5-state variable thus has one of the states 0, 1, 2, 3, 4. These states are presented in sequence as [0 1 2 3 4]. The value of a state in the presentation is also its position in [0 1 2 …. (n-1)]. The representation is thus in origin 0. Each *n*-state inverter has elements selected from [0 1 2 … (n-1)].

An inverter inverts or transforms. For instance a 5-state inverter may be:
[0 1 2 3 4]→[1 2 3 4 0]. The sequence after the arrow '→' indicates the result of the inversion. The inversion result in position 0 of [1 2 3 4 0] is 1; the inversion result in position 1 of [1 2 3 4 0] is 2; and so on with the inversion result in position 4 of [1 2 3 4 0] is 0.

The inverter [*a b c d e*] can be interpreted as: 0→*a*; 1→*b*; 2→*c*; 3→*d*; and 4→*e* of the 5-state inverter [0 1 2 3 4] →[*a b c d e*]. All elements *a*, *b*, *c*, *d*, and *e* are selected from 0, 1, 2, 3, and 4. For an *n*-state inverter to be reversible, no duplicates of a state may occur. The inverter [0 1 2 3 4] →[0 1 2 3 4] is called identity as it inverts onto itself.

Each reversible inverter has a reversing inverter. A reversible inverter followed by its reversing inverter creates identity. As an example: the 5-state inverter [0 1 2 3 4] →[2 3 4 0 1] has as reversing inverter [0 1 2 3 4] →[3 4 0 1 2]. The 5-state inverter [0 1 2 3 4] →[4 2 1 3 0] is self-reversing as it has itself as a reversing inverter.

The FLT is characterized by the following figure:



A device performs an operation in accordance with definition '*muln*,' such as a multiplication modulo-*n*, upon two inputs '*a*' and '*c*' and with output '*d*.' Input state *a* is result of an inversion of state *i*1 by inverter *inv*1. Input state *c* is result of an inversion of state *i*2 by inverter *inv*1. Furthermore, output state *d* is inverted by inverter *rinv*1, which is the reversing inverter of *inv*1. The above device performs an *n*-state Finite Lab-Transform (FLT) that preserves meta-properties of operation *muln*.

The following 5-state switching tables show the FLT of a computer operation *mul*5 characterized by a modulo-5 multiplication in accordance with inverter [0 1 2 3 4] →[2 3 4 0 1] to create *mul*5*a*.

*mul*5	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

*mul*5*a*	0	1	2	3	4
0	2	4	1	3	0
1	4	2	0	3	1
2	1	0	4	3	2
3	3	3	3	3	3
4	0	1	2	3	4

The zero-element of *mul*5 is $z=0$ and the one-element $e=1$. The operation is commutative and associative. The zero-element of *mul*5*a* is $z=3$ and its one-element $e=4$. The operation is clearly commutative. It is also associative, which is easy to check. Thus, operation *mul*5*a* can be used for exponentiation. The result is a customization of computer operation *mul*5.

Actually any *n*-state computer operation can be modified and customized by an FLT, including additions and/or multiplications over GF($n=2^p$) as used in AES, SHA, RSA and ECC.

The FLT by itself is believed to be a novel number-theoretical concept, as being a property preserving transformation. It is applied in computer operation design to create novel computer functions. A toy example applying the FLT in Diffie-Hellman Key Exchange can be found here.

Mathematical Logic and Machine Logic

People often confuse Mathematical Logic with Machine Logic. Mathematical logic is the description of a calculus, such as Boolean Algebra. Machine Logic is a description of physical behavior of a computer circuit by using Mathematical Logic. How these concepts mesh requires a fairly elaborate explanation. The best understanding is provided by using the distinguishing elements of a computer design in 3 levels: 1) architecture (what the user sees); 2) implementation (the logic description); 3) the physical realization. Levels 1 and 2 find their roots in level 3. Further details can be found in the work of my teacher Prof. Dr. G.A. Blaauw, who with Dr. Fred Brooks and Dr. Gene Amdahl, was a co-architect of the legendary IBM System/360.

Peter Lablans, January 23, 2020

Contact: ip@ternarylogic.com